HW 1

AUTHOR
Shijie An

Homework 1

Use the jcpd-calls-for-service.csv file found on Canvas for the following exercise. The file contains data on the service calls received by the Jersey City NJ police department.

Part 1 - Reading and Manipulating Data

1. Read the data from the file into a data frame called jcpd

```
# jcpd <- read.csv("D:/CU TC/hudk 4050 data mining/HUDK_4050_data_mining_HW/jcpd-calls-for-se
# path <- paste("D:/CU TC/hudk 4050 data mining/",
#               "HUDK_4050_data_mining_HW/jcpd-calls-for-service.csv",
#               sep = "")
# jcpd <- read.csv(path)
path2 <- paste(getwd(),
               "/jcpd-calls-for-service.csv",
               sep = "")
# path2 <- "jcpd-calls-for-service.csv"
jcpd <- read.csv(path2)
```

2. Find the number of rows and columns in the data and inspect the data by printing out the first and last few rows

```
## 22 variables (columns), 107331 rows
summary(jcpd)
```

```
event.number       time.received      time.dispatched    time.arrived
Length:107331      Length:107331      Length:107331      Length:107331
Class :character   Class :character   Class :character   Class :character
Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
  callcode         call.code.description  call.type         is.primary
Length:107331      Length:107331          Length:107331      Mode :logical
Class :character   Class :character       Class :character   FALSE:28313
Mode  :character   Mode  :character       Mode  :character   TRUE :79018
```

```
  address              City              geo.count       geo.error
Length:107331      Length:107331      Min.   :-1.000   Length:107331
Class :character   Class :character   1st Qu.: 1.000   Class :character
Mode  :character   Mode  :character   Median : 1.000   Mode  :character
                                      Mean   : 1.032
                                      3rd Qu.: 1.000
                                      Max.   : 3.000
                                      NA's   :55
Call.Category      Description        Category.Code      Coordinates
Length:107331      Length:107331      Length:107331      Length:107331
Class :character   Class :character   Class :character   Class :character
Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
   priority     Time.Elapsed          Injury            district
Min.   :1.000   Length:107331      Length:107331      Length:107331
1st Qu.:3.000   Class :character   Class :character   Class :character
Median :6.000   Mode  :character   Mode  :character   Mode  :character
Mean   :5.481
3rd Qu.:9.000
Max.   :9.000

    shift        unit.id
Min.   :1.00   Length:107331
1st Qu.:2.00   Class :character
Median :2.00   Mode  :character
Mean   :2.26
3rd Qu.:3.00
Max.   :3.00
```

```
# jcpd[1, ]
head(jcpd)
```

```
  event.number             time.received           time.dispatched
1    17-139705 2017-07-10 23:08:00-04:00 2017-07-10 23:09:00-04:00
2    17-141026 2017-07-12 19:02:00-04:00 2017-07-12 19:03:00-04:00
3    17-063896 2017-04-02 17:54:00-04:00 2017-04-02 17:56:00-04:00
4    17-125297 2017-06-22 22:34:00-04:00 2017-06-22 22:36:00-04:00
5    17-063939 2017-04-02 18:54:00-04:00 2017-04-02 19:03:00-04:00
6    17-125351 2017-06-23 00:18:00-04:00 2017-06-23 00:19:00-04:00
              time.arrived callcode
1 2017-07-10 23:08:00-04:00   J17.00
2 2017-07-12 19:02:00-04:00   J17.00
3 2017-04-02 18:11:00-04:00   D11.00
```

```
5 2017-04-02 18:11:00-04:00   D11.00
4 0001-01-01 00:03:58-04:56   C31.00
5 0001-01-01 00:03:58-04:56   H19.00
6 2017-06-23 00:54:00-04:00   B23.20
                                                                                call.code.description
1 DISTRICT MISSION / ASSIGNMENT; TIME ELAPSED NOT APPLICABLE; INJURY NOT APPLICABLE
2 DISTRICT MISSION / ASSIGNMENT; TIME ELAPSED NOT APPLICABLE; INJURY NOT APPLICABLE
3        MOTOR VEHICLE ACCIDENT; TIME ELAPSED NOT APPLICABLE; INJURY NOT APPLICABLE
4    BURG ALARM COMMERCIAL PROP; TIME ELAPSED NOT APPLICABLE; INJURY NOT APPLICABLE
5                     CC CITIZEN; TIME ELAPSED NOT APPLICABLE; INJURY NOT APPLICABLE
6    OTHER INTER/CONFLICT; IN PROGRESS AND/OR ACTOR ON SCENE; INJURY NOT APPLICABLE
  call.type is.primary                address        City geo.count  geo.error
1        SI       TRUE            207 7TH ST; JERSEY CITY         1 S5HPNTSCZA
2        SI       TRUE            207 7TH ST; JERSEY CITY         1 S5HPNTSCZA
3        PH       TRUE BERGEN AVE & UNION ST; JERSEY CITY         1         SX
4        PH       TRUE    100 CAVEN POINT RD; JERSEY CITY         1 S5HPNTSCZA
5       911       TRUE    29 WESTERVELT PL; 2 JERSEY CITY         1 S5HPNTSCZA
6        PH       TRUE       341 MONMOUTH ST; JERSEY CITY         1 S5HPNTSCZA
                   Call.Category                   Description Category.Code
1 Administrative Police Activity District Mission / Assignment             J
2 Administrative Police Activity District Mission / Assignment             J
3       Vehicle/Traffic Problems        Motor Vehicle Accident             D
4                Property Crimes    Burg Alarm Commercial Prop             C
5                     Assistance                    Cc Citizen             H
6         Interpersonal Conflict          Other Inter/conflict             B
                Coordinates priority                      Time.Elapsed
1 40.72590637, -74.04273987        9       Time Elapsed Not Applicable
2 40.72590637, -74.04273987        9       Time Elapsed Not Applicable
3 40.71577072, -74.07904053        3       Time Elapsed Not Applicable
4  40.69612503, -74.0770874        4       Time Elapsed Not Applicable
5 40.71648026, -74.06636047        8       Time Elapsed Not Applicable
6 40.72100449, -74.05013275        2 In Progress And/or Actor On Scene
                Injury district shift unit.id
1 Injury Not Applicable     East     3    E203
2 Injury Not Applicable     East     3    E503
3 Injury Not Applicable     West     3    W303
4 Injury Not Applicable    South     3    S303
5 Injury Not Applicable     East     3    E103
6 Injury Not Applicable     East     1    E301
```

```
tail(jcpd)
```

```
       event.number             time.received           time.dispatched
107326    17-025931 2017-02-07 17:05:00-05:00 2017-02-07 17:07:00-05:00
107327    17-060671 2017-03-28 22:09:00-04:00 2017-03-28 22:10:00-04:00
107328    17-043687 2017-03-04 10:35:00-05:00 2017-03-04 10:39:00-05:00
107329    17-041679 2017-03-01 15:12:00-05:00 2017-03-01 15:13:00-05:00
107330    17-025993 2017-02-07 18:42:00-05:00 2017-02-07 18:43:00-05:00
107331    17-026003 2017-02-07 18:51:00-05:00 2017-02-07 18:52:00-05:00
                   time.arrived callcode
107326 2017-02-07 17:05:00-05:00   J24.00
```

```
107326 2017-02-07 17:09:00-05:00   J24.00
107327 2017-03-28 22:09:00-04:00   D16.00
107328 2017-03-04 10:50:00-05:00   F21.00
107329 2017-03-01 15:13:00-05:00   E14.00
107330 2017-02-07 18:42:00-05:00   J14.00
107331 2017-02-07 18:57:00-05:00   G18.00

call.code.description
107326                      DIRECTED PATROL; TIME ELAPSED NOT APPLICABLE; INJURY NOT
APPLICABLE
107327 SIGN DOWN/TRAFFIC LIGHT OUT OF ORDER; TIME ELAPSED NOT APPLICABLE; INJURY NOT
APPLICABLE
107328                   SLUMPER IN VEHICLE; TIME ELAPSED NOT APPLICABLE; INJURY NOT
APPLICABLE
107329          STREET FIGHT ( NO WEAPONS ); TIME ELAPSED NOT APPLICABLE; INJURY NOT
APPLICABLE
107330                                MEAL; TIME ELAPSED NOT APPLICABLE; INJURY NOT
APPLICABLE
107331                  FIREARM PERSON WITH; TIME ELAPSED NOT APPLICABLE; INJURY NOT
APPLICABLE
       call.type is.primary                                  address
107326        SI       TRUE OLD BERG (OLD BERGEN RD) & DANFORTH AVE;
107327        SI       TRUE         NEWARK AVE & US HIGHWAY 1 AND 9;
107328        PH       TRUE                              100 SKYWAY;
107329        SI       TRUE                          60 CRESCENT AVE;
107330        SI      FALSE                             1 JACKSON ST;
107331        PH      FALSE        737 WESTSIDE AVE (WEST SIDE AVE);
             City geo.count  geo.error                      Call.Category
107326 JERSEY CITY         1         SX     Administrative Police Activity
107327 JERSEY CITY         0  Not Found           Vehicle/Traffic Problems
107328 JERSEY CITY         0  Not Found Suspicious/Hazardous Circumstances
107329 JERSEY CITY         1 S5HPNTSCZA                    Public Nuisance
107330 JERSEY CITY         1 S5HPN-SC-A     Administrative Police Activity
107331 JERSEY CITY         1 S5HPNTSC-A
                              Description Category.Code
107326                      Directed Patrol             J
107327 Sign Down/traffic Light Out Of Order             D
107328                   Slumper In Vehicle             F
107329          Street Fight ( No Weapons )             E
107330                                Meal              J
107331                  Firearm Person With             G
                           Coordinates priority               Time.Elapsed
107326        40.69797134, -74.09262848        9 Time Elapsed Not Applicable
107327                                         7 Time Elapsed Not Applicable
107328                                         2 Time Elapsed Not Applicable
107329 40.717857359999996, -74.07035828        3 Time Elapsed Not Applicable
107330        40.71671677, -74.07403564        9 Time Elapsed Not Applicable
107331        40.72566223, -74.07739258        2 Time Elapsed Not Applicable
                     Injury district shift unit.id
107326 Injury Not Applicable    South     3    SSD2
107327 Injury Not Applicable    North     3    N403
```

```
107328 Injury Not Applicable     West     2    W302
107329 Injury Not Applicable     West     3   WLHS1
107330 Injury Not Applicable     West     3    W303
107331 Injury Not Applicable     West     3  WINV35
```

3. How many missing values are there? How many rows are there with missing values?

```
## num of missing values
sum(is.na(jcpd))
```

```
[1] 55
```

```
## num of rows with missing values
sum(!complete.cases(jcpd))
```

```
[1] 55
```

4. Find which columns have the missing values. Hint: Use the which() function.

```
## the address of num of columns with missing values
missing <- which(is.na(jcpd))
# so if we have the location, we will use this location to devide the row num.
# we take the floor of deviation, and plus 1, so we can get col.
# since we have 107331 cols, we will use the location devided by 107331 and + 1.
# missingCol <- floor(missing / nrow(jcpd)) + 1
missingCol <- floor(missing / 107331) + 1
missingCol
```

```
 [1] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
[26] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
[51] 11 11 11 11 11
```

```
unique(missingCol)
```

```
[1] 11
```

5. Notice that the column names have spaces in them. This is not a valid column name in R. So, replace all the spaces in the names with an underscore to comply with the snake_case convention. Hint: The names() function returns column names.

```
old_name <- colnames(jcpd)
new_name <- gsub("[.]", "_", old_name)
names(jcpd) <- new_name
colnames(jcpd)
```

```
[1] "event_number"          "time_received"         "time_dispatched"
[4] "time_arrived"          "callcode"              "call_code_description"
```

```
 [7] "call_type"            "is_primary"           "address"
[10] "City"                 "geo_count"            "geo_error"
[13] "Call_Category"        "Description"          "Category_Code"
[16] "Coordinates"          "priority"             "Time_Elapsed"
[19] "Injury"               "district"             "shift"
[22] "unit_id"
```

6. Replace the missing values in the column geo_count with the number 1

```
# num = ""
# i <- 1
# for(num in jcpd$geo_count) {
#     if(is.na(num)) {
#         jcpd$geo.count[i] = 1
#    }
#    i <- i + 1
# }
# jcpd$geo_count
  jcpd$geo_count[is.na(jcpd$geo_count)] <- 1
  jcpd$geo_count[is.na(jcpd$geo_count)]
```

```
numeric(0)
```

7. Remove all the remaining missing values from the data frame

```
nomissing <- na.omit(jcpd)
nomissing
```

8. Check to see if there are any duplicate rows and if there are any remove them using the pipe operator and the dplyr function distinct()

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
sum(duplicated(jcpd))
```

```
[1] 0
```

```
jcpd %>%
  distinct(.keep_all = TRUE)
```

9. Sort the data by descending call type

```
library(dplyr)
jcpd %>%
  arrange(desc(jcpd$call_type))
```

10. Create a new data frame called jcpd911 by filtering the original dataset for the 911 calls. Print out the first six rows and check if the filtering worked. How many 911 calls were there?

```
jcpd911 <-  jcpd %>%
  filter(call_type == 911)
head(jcpd911)
```

11. Create a new variable (column) called dispatch_duration in the jcpd dataset by subtracting time_received from time_dispatched. Hint: You also need to convert the format of time.received and time.dispatched using the strptime() command and then subtract.

```
receive <- strftime(jcpd$time_received, "%y-%m-%d %H:%M:%S")
dispatch <- strftime(jcpd$time_dispatched, "%y-%m-%d %H:%M:%S")
new_receive <- strptime(receive, "%y-%m-%d %H:%M:%S")
new_dispatch <- strptime(dispatch, "%y-%m-%d %H:%M:%S")
dispatch_duration <- new_dispatch - new_receive
# dispatch_duration
jcpd <- jcpd %>%
  mutate(dispatch_duration)
```

12. Now check if there are missing values in the newly created column and also check for dispatch durations that are negative or zero. This is garbage data so remove these rows.

```
sum(is.na(jcpd$dispatch_duration))
```

```
[1] 0
```

```
delete <- which(jcpd$dispatch_duration <= 0)
jcpd <- jcpd[-c(delete), ]
jcpd
```

13. Find the average (mean) dispatch duration using the new variable (column) you created above

```
mean <- mean(jcpd$dispatch_duration)
mean
```

```
Time difference of 125.4569 secs
```

14. Find the average (mean) dispatch duration by call type

```
unique(jcpd$call_type)
```

```
[1] "SI"  "PH"  "911" "MVS" "STA" "IC"
```

```
library(dplyr)
# I will use filter to calculate the differnt mean for different call type
# SI <- jcpd %>%
#   filter(call_type == "SI")
# SI_mean <- mean(SI$dispatch_duration)
#
# PH <- jcpd %>%
#   filter(call_type == "PH")
# PH_mean <- mean(PH$dispatch_duration)
#
# C911 <- jcpd %>%
#   filter(call_type == "911")
# mean_911 <- mean(C911$dispatch_duration)
#
# MVS <- jcpd %>%
#   filter(call_type == "MVS")
# MVS_mean <- mean(MVS$dispatch_duration)
#
# STA <- jcpd %>%
#   filter(call_type == "STA")
# STA_mean <- mean(STA$dispatch_duration)
#
# IC <- jcpd %>%
#   filter(call_type == "IC")
# IC_mean <- mean(IC$dispatch_duration)
jcpd %>%
  group_by(call_type) %>%
  summarize(mean(dispatch_duration))
```

```
# A tibble: 6 × 2
  call_type `mean(dispatch_duration)`
  <chr>     <drtn>
1 911       160.96279 secs
2 IC        300.00000 secs
3 MVS        85.76130 secs
4 PH        142.79056 secs
5 SI         77.98542 secs
```

```
6 STA       144.00000 secs
```

15. How many rows contain the word GUNSHOTS in the call code description column. Hint: Use the stringr package in tidyverse. Use the str_detect function in stringr. Use help to learn about the function.

```
library(stringr)
sum(str_detect(jcpd$call_code_description, "GUNSHOTS") == TRUE)
```

```
[1] 1334
```

16. Now create a data frame called jcpd_gunshots that has just the rows that contain the word GUNSHOTS in the call code description

```
jcpd_gunshots_row <- which(str_detect(jcpd$call_code_description, "GUNSHOTS") == TRUE)
jcpd_gunshots <- jcpd[jcpd_gunshots_row, ]
jcpd_gunshots
```

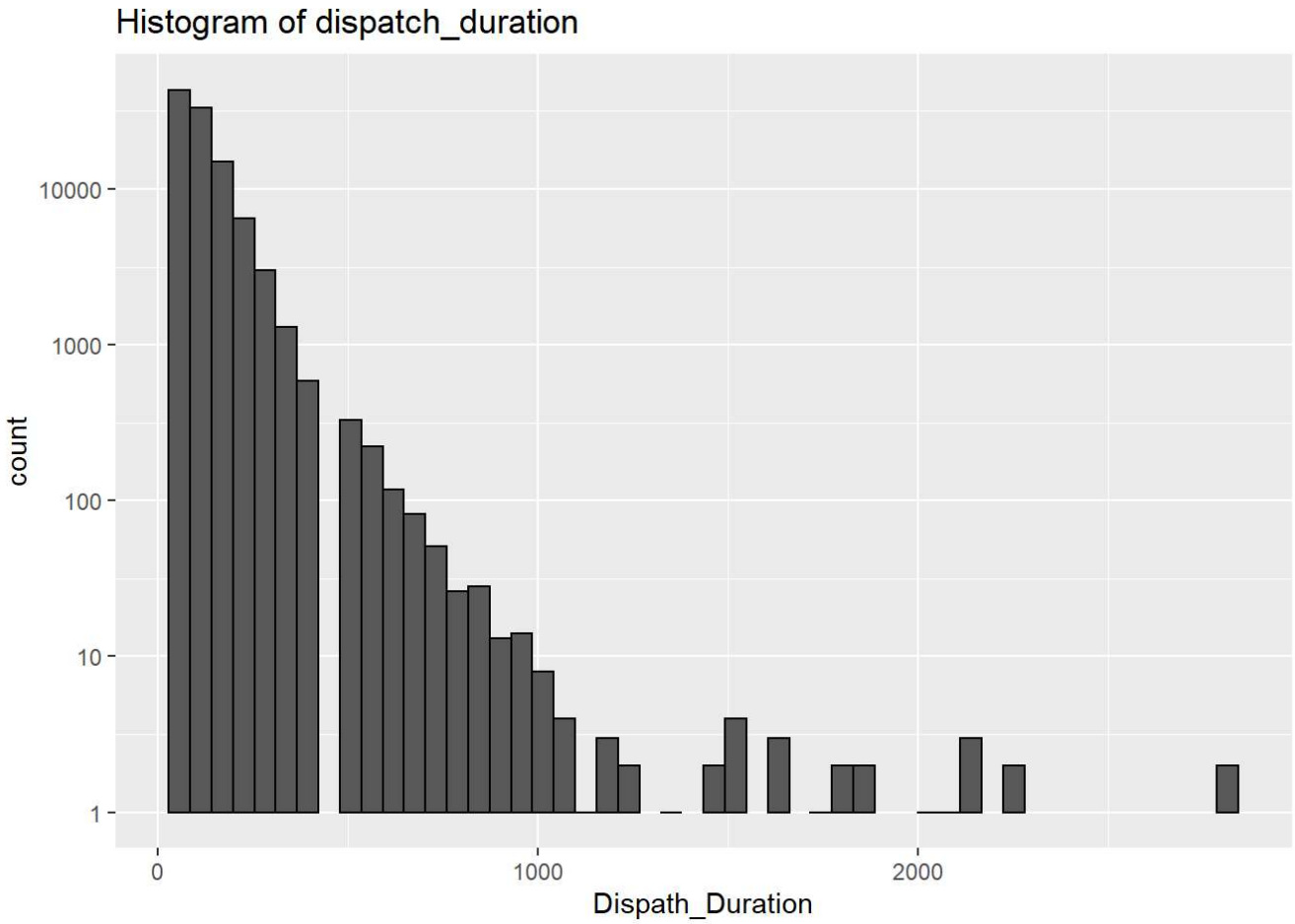
Part 2 - Plotting

Use ggplot2 for all the questions

1. Plot a histogram of dispatch duration. What can you infer from the histogram?

```
library(dplyr)
library(ggplot2)
ggplot(jcpd, aes(x = dispatch_duration)) +
  geom_histogram(bins = 50, col = "black") +
  scale_y_log10() +
  xlab("Dispath_Duration") +
  ggtitle("Histogram of dispatch_duration")
```

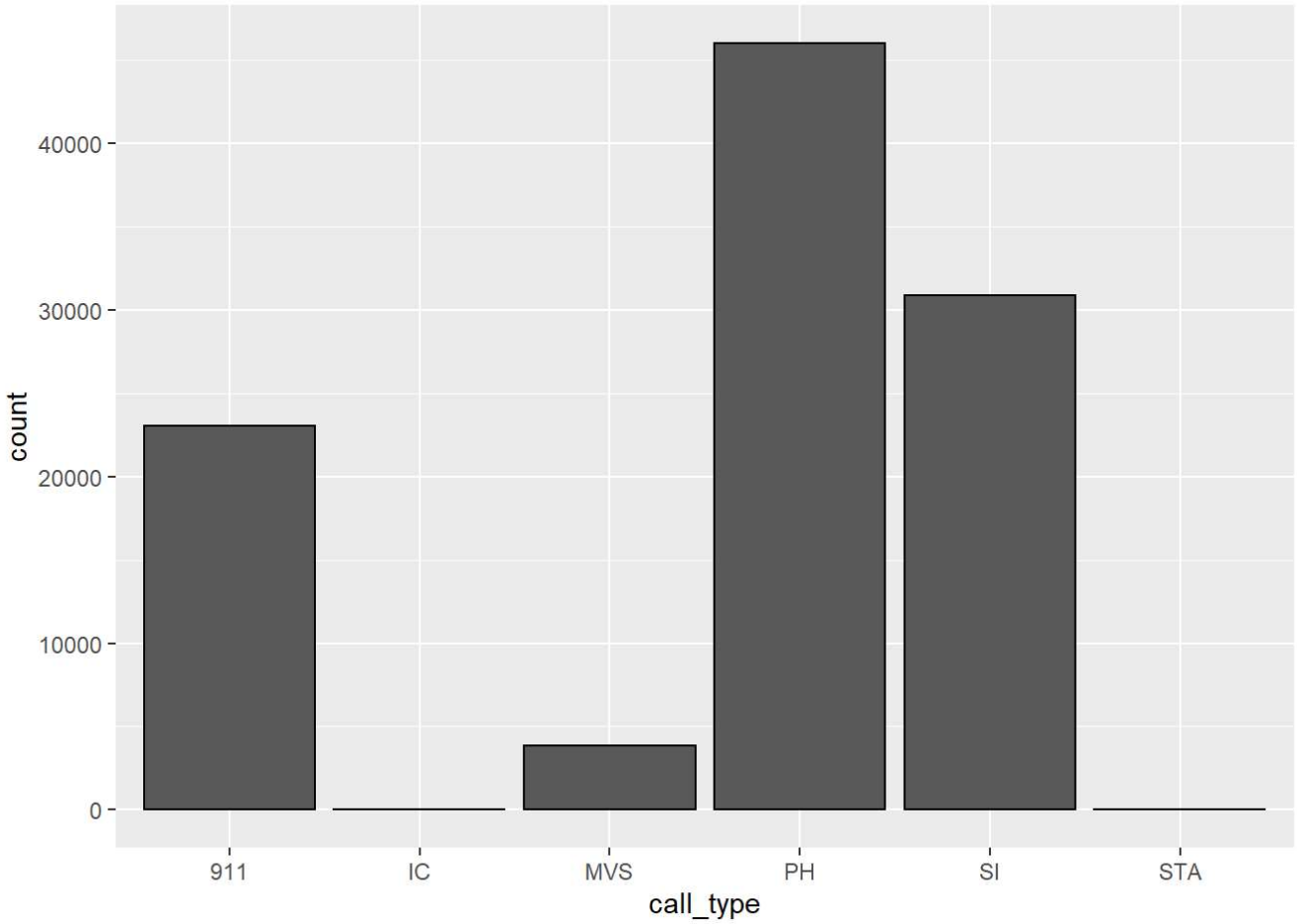
```
Don't know how to automatically pick scale for object of type difftime. Defaulting to
continuous.

Warning: Transformation introduced infinite values in continuous y-axis

Warning: Removed 17 rows containing missing values (geom_bar).
```



It can be inferred that the histogram plot the frequency and counts the time that each number appears. It showed that 60 appear the most of times, it reaches to 40000 times. and the max number appear one time and is so big that reaches to 3000.

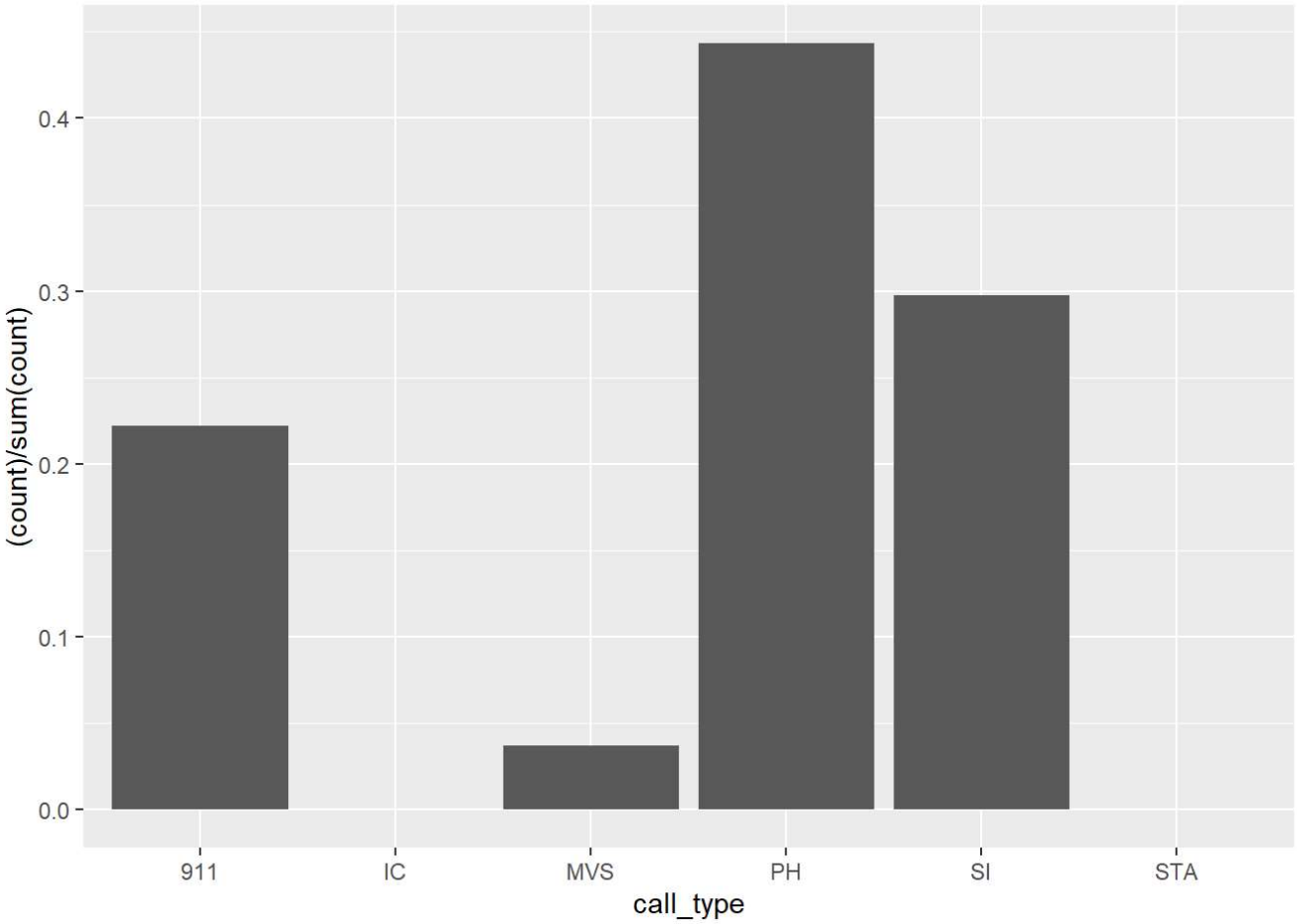
2. Draw a bar chart of the count of calls by call type.

```
count <- jcpd %>%
  group_by(call_type)

ggplot(count, aes(x = call_type)) +
  geom_bar(col = "black")
```



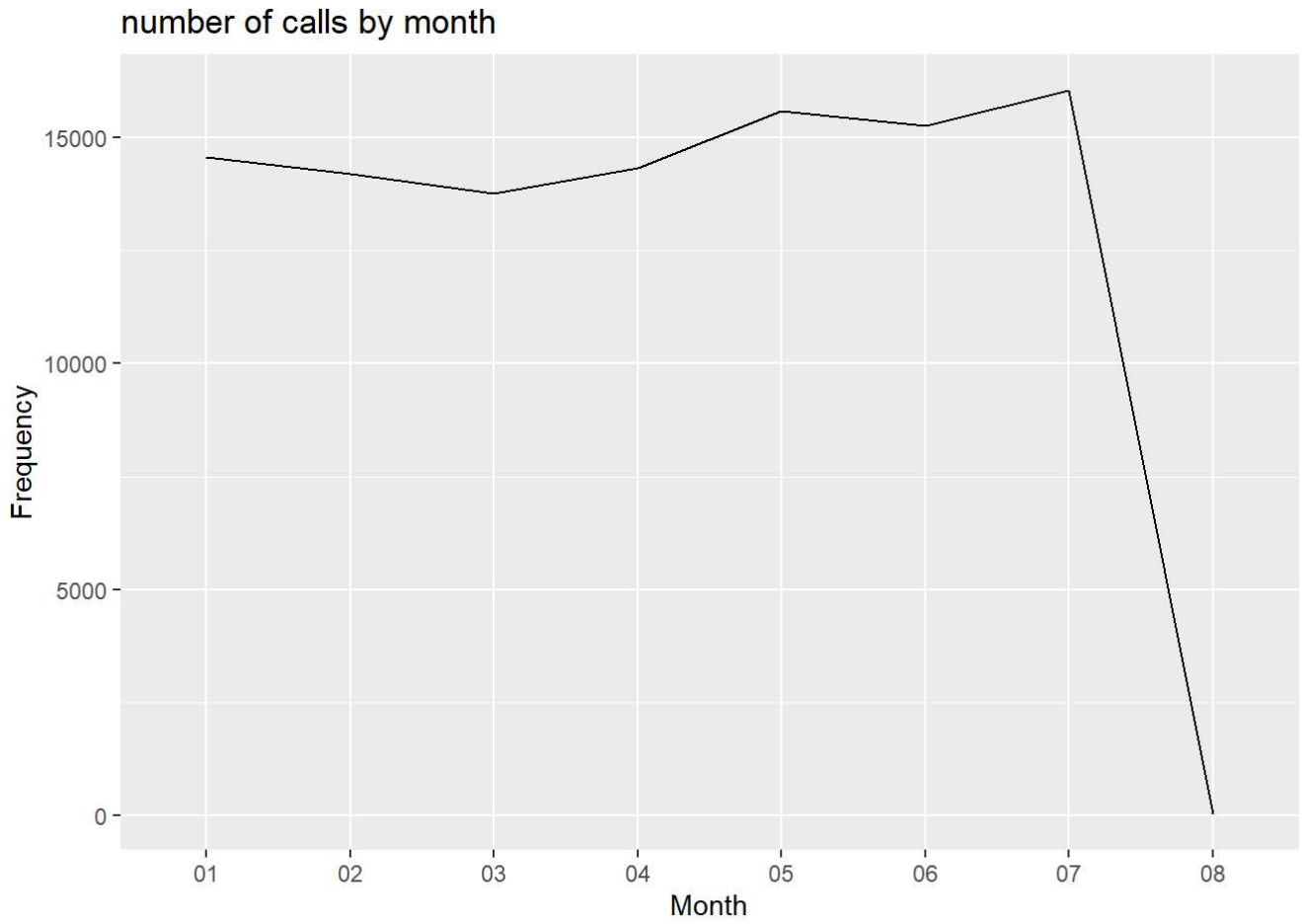
3. Draw a bar chart of the proportion of calls by call type

```
# jcpd_test %>%
#   group_by(call_type) %>%
#   mutate(freq = )
ggplot(count, aes(x = call_type)) +
  geom_bar(aes(y = (..count..) / sum(..count..)))
```



4. Create new column called call_month in the jcpd data frame and store the month extracted from the time_received column. Plot the number of calls by month as a line graph.

```
call_month <- strftime(jcpd$time_received, "%m")

jcpd <- jcpd %>%
  mutate(call_month)

# num_call <- jcpd %>%
#   group_by(call_month) %>%
#   summarize()

# ggplot

# ggplot(num_call, aes(x = call_month)) +
#   geom_bar()
# made line 跑不了，只有bar能计算频数

table <- as.data.frame(table(jcpd$call_month))
ggplot(table, aes(x = Var1, y = Freq, group = 1)) +
  geom_line() +
  xlab("Month") +
  ylab("Frequency") +
  ggtitle("number of calls by month")
```



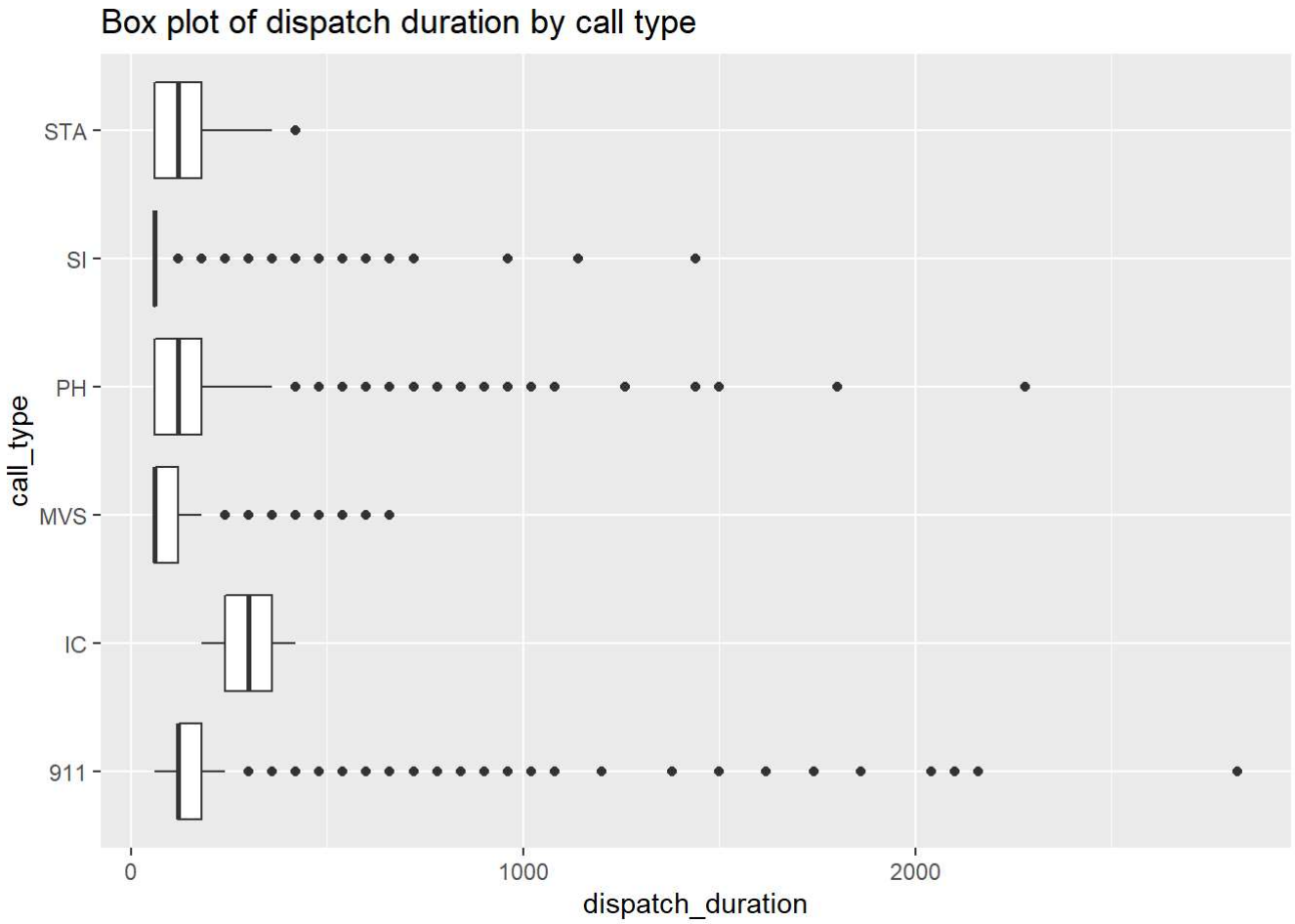
5. Plot a box and whiskers plot of the dispatch duration by call type. Flip the coordinates so the call type appears on the Y axis. Give the plot a title - "Box plot of dispatch duration by call type"

```
count_sum <- jcpd %>%
  group_by(call_type) %>%
  summarize(dispatch_duration)
```

```
`summarise()` has grouped output by 'call_type'. You can override using the
`.groups` argument.
```

```
ggplot(count_sum, aes(x = dispatch_duration, y = call_type)) +
  geom_boxplot() +
  ggtitle("Box plot of dispatch duration by call type")
```

```
Don't know how to automatically pick scale for object of type difftime. Defaulting to continuous.
```



Part 3 - Joins

Use dplyr for all joins

1. Load the library nycflights13 after installing it.

```
# install.packages("nycflights13")
library("nycflights13")
planes
```

```
# A tibble: 3,322 × 9
   tailnum  year type                  manuf…¹ model engines seats speed engine
   <chr>   <int> <chr>                 <chr>   <chr>   <int> <int> <int> <chr>
 1 N10156   2004 Fixed wing multi engi… EMBRAER EMB-…       2    55    NA Turbo…
 2 N102UW   1998 Fixed wing multi engi… AIRBUS… A320…       2   182    NA Turbo…
 3 N103US   1999 Fixed wing multi engi… AIRBUS… A320…       2   182    NA Turbo…
 4 N104UW   1999 Fixed wing multi engi… AIRBUS… A320…       2   182    NA Turbo…
 5 N10575   2002 Fixed wing multi engi… EMBRAER EMB-…       2    55    NA Turbo…
 6 N105UW   1999 Fixed wing multi engi… AIRBUS… A320…       2   182    NA Turbo…
 7 N107US   1999 Fixed wing multi engi… AIRBUS… A320…       2   182    NA Turbo…
 8 N108UW   1999 Fixed wing multi engi… AIRBUS… A320…       2   182    NA Turbo…
 9 N109UW   1999 Fixed wing multi engi… AIRBUS… A320…       2   182    NA Turbo…
```

```
10 N110UW   1999 Fixed wing multi engi… AIRBUS… A320…       2   182    NA Turbo…
# … with 3,312 more rows, and abbreviated variable name ¹manufacturer
```

2. What is the primary key of the planes table?

   tailnum is the primary key of the planes table

3. Add full airline name from the airlines data frame to the flights data frame and create a new data frame called flights_with_names.

```
flights_with_names <- full_join(nycflights13::airlines, nycflights13::flights, by = "carrier")
flights_with_names
```

```
# A tibble: 336,776 × 20
   carrier name         year month   day dep_t…¹ sched…² dep_d…³ arr_t…⁴ sched…⁵
   <chr>   <chr>       <int> <int> <int>   <int>   <int>   <dbl>   <int>   <int>
 1 9E      Endeavor A…  2013     1     1     810     810       0    1048    1037
 2 9E      Endeavor A…  2013     1     1    1451    1500      -9    1634    1636
 3 9E      Endeavor A…  2013     1     1    1452    1455      -3    1637    1639
 4 9E      Endeavor A…  2013     1     1    1454    1500      -6    1635    1636
 5 9E      Endeavor A…  2013     1     1    1507    1515      -8    1651    1656
 6 9E      Endeavor A…  2013     1     1    1530    1530       0    1650    1655
 7 9E      Endeavor A…  2013     1     1    1546    1540       6    1753    1748
 8 9E      Endeavor A…  2013     1     1    1550    1550       0    1844    1831
 9 9E      Endeavor A…  2013     1     1    1552    1600      -8    1749    1757
10 9E      Endeavor A…  2013     1     1    1554    1600      -6    1701    1734
# … with 336,766 more rows, 10 more variables: arr_delay <dbl>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>, and abbreviated variable names
#   ¹dep_time, ²sched_dep_time, ³dep_delay, ⁴arr_time, ⁵sched_arr_time
```

4. Add the destination latitude and longitude by joining the flights_with_names data frame and the airports data frame

```
join_lat_lon <- flights_with_names %>%
  left_join(airports, by = c("origin" = "faa"))
join_lat_lon
```

```
# A tibble: 336,776 × 27
   carrier name.x       year month   day dep_t…¹ sched…² dep_d…³ arr_t…⁴ sched…⁵
   <chr>   <chr>       <int> <int> <int>   <int>   <int>   <dbl>   <int>   <int>
 1 9E      Endeavor A…  2013     1     1     810     810       0    1048    1037
 2 9E      Endeavor A…  2013     1     1    1451    1500      -9    1634    1636
 3 9E      Endeavor A…  2013     1     1    1452    1455      -3    1637    1639
 4 9E      Endeavor A…  2013     1     1    1454    1500      -6    1635    1636
 5 9E      Endeavor A…  2013     1     1    1507    1515      -8    1651    1656
 6 9E      Endeavor A…  2013     1     1    1530    1530       0    1650    1655
 7 9E      Endeavor A…  2013     1     1    1546    1540       6    1753    1748
 8 9E      Endeavor A…  2013     1     1    1550    1550       0    1844    1831
```

```
 9 9E      Endeavor A…  2013     1     1    1552    1600      -8    1749    1757
10 9E      Endeavor A…  2013     1     1    1554    1600      -6    1701    1734
# … with 336,766 more rows, 17 more variables: arr_delay <dbl>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>, name.y <chr>, lat <dbl>,
#   lon <dbl>, alt <dbl>, tz <dbl>, dst <chr>, tzone <chr>, and abbreviated
#   variable names ¹dep_time, ²sched_dep_time, ³dep_delay, ⁴arr_time,
#   ⁵sched_arr_time
```

5. Compute the average delay by destination in the flights data frame and store in a new data frame called delays. Now join the latitude and longitude information from the airports data frame to the delays data frame.

```
flights <- flights
sum(is.na(flights$arr_delay))
```

```
[1] 9430
```

```
flights$arr_delay[is.na(flights$arr_delay)] <- 0

delays <- flights %>%
  group_by(dest) %>%
  summarize(flights_delay = mean(arr_delay))
delays
```

```
# A tibble: 105 × 2
   dest  flights_delay
   <chr>         <dbl>
 1 ABQ            4.38
 2 ACK            4.83
 3 ALB           13.7
 4 ANC           -2.5
 5 ATL           11.1
 6 AUS            5.95
 7 AVL            7.60
 8 BDL            6.56
 9 BGR            7.66
10 BHM           15.3
# … with 95 more rows
```

6. Create a data frame called top_dest_delay that has the destinations with top five delay times

```
top_dest_delay <- flights %>%
  arrange(desc(arr_delay)) %>%
  head(n = 5)
top_dest_delay
```

```
# A tibble: 5 × 19
```

```
   year month   day dep_time sched_dep…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵ carrier
  <int> <int> <int>    <int>       <int>   <dbl>   <int>   <int>   <dbl> <chr>
1  2013     1     9      641         900    1301    1242    1530    1272 HA
2  2013     6    15     1432        1935    1137    1607    2120    1127 MQ
3  2013     1    10     1121        1635    1126    1239    1810    1109 MQ
4  2013     9    20     1139        1845    1014    1457    2210    1007 AA
5  2013     7    22      845        1600    1005    1044    1815     989 MQ
# … with 9 more variables: flight <int>, tailnum <chr>, origin <chr>,
#   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
#   time_hour <dttm>, and abbreviated variable names ¹sched_dep_time,
#   ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
```

7. Now filter the flights table to contain only records for the top five destinations you found in the previous question and create a new data frame called flight_delay_top. Hint: You can use a semi_join to do this

```
top_dest_delay <- semi_join(flights, top_dest_delay, by = "dest")
top_dest_delay
```

```
# A tibble: 38,786 × 19
   year month   day dep_time sched_de…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵ carrier
  <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
 1  2013     1     1      554        558      -4     740     728      12 UA
 2  2013     1     1      558        600      -2     753     745       8 AA
 3  2013     1     1      558        600      -2     923     937     -14 UA
 4  2013     1     1      608        600       8     807     735      32 MQ
 5  2013     1     1      611        600      11     945     931      14 UA
 6  2013     1     1      629        630      -1     824     810      14 AA
 7  2013     1     1      655        700      -5    1037    1045      -8 DL
 8  2013     1     1      656        700      -4     854     850       4 AA
 9  2013     1     1      709        700       9     852     832      20 UA
10  2013     1     1      715        713       2     911     850      21 UA
# … with 38,776 more rows, 9 more variables: flight <int>, tailnum <chr>,
#   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#   minute <dbl>, time_hour <dttm>, and abbreviated variable names
#   ¹sched_dep_time, ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
```